NORTH-HOLLAND

Solving Linear Systems Involved in Constrained Optimization

Yixun Shi
Department of Mathematics & Computer Science
Bloomsburg University of Pennsylvania
Bloomsburg, Pennsylvania 17815

Submitted by Richard A. Brualdi

ABSTRACT

Many interior point methods for large scale linear programming, quadratic programming, the convex programming solve an $(n + m) \times (n + m)$ linear system in each iteration. The last m equations require exact solutions in order to maintain the feasibility. Current implementations reduce that step to solving an $m \times m$ linear system. The solution must be exact, because otherwise the error would be entirely passed on to the last m equations of the original system. This makes the computation costly and sometimes impractical. In this paper, we propose an inexpensive iterative method for solving that $(n + m) \times (n + m)$ system. It guarantees exact solutions to the last m equations. The convergence is proved, and the implementational issues are discussed. Some preliminary numerical results are also reported.

1. INTRODUCTION

An $(n + m) \times (n \times m)$ linear system of the form

$$\begin{aligned} Dx - A^T y &= b, \\ Ax \qquad\quad &= 0 \end{aligned} \tag{1}$$

needs to be solved in each iteration of many interior point methods for large scale linear programming, quadratic programming, and convex programming,

LINEAR ALGEBRA AND ITS APPLICATIONS 229:175–189 (1995)

655 Avenue of the Americas, New York, NY 10010

0024-3795/95/$9.50
SSDI 0024-3795(94)00007-Z

where $D \in R^{n\times n}$ is either a positive diagonal matrix or the sum of a positive diagonal matrix and a symmetric positive semidefinite matrix, $A \in R^{m\times n}$, $m < n$, Rank $A = m$, $b \in R^n$, and $0 = (0, 0, \ldots, 0)^T \in R^m$. The matrices D and b will change in each iteration, while A, as the original constraint matrix, stays the same. To see just a few examples among many others, we list the following cases.

Case 1 (linear programming). Y. Ye (1992) describes a primal-dual interior point algorithm for solving the linear programming problem

$$\begin{aligned} &\min \quad c^T x \\ &\text{s.t.} \quad Ax = b, \quad x \geqslant 0 \end{aligned} \tag{2}$$

where $A \in R^{m\times n}$, $m < n$, Rank $A = m$, $c \in R^n$, $x \in R^n$, and $b \in R^m$. The algorithm is as follows.

ALGORITHM 1.1 (Y. Ye, 1992). *Take $x_0 \in R^n$ such that $Ax_0 = b$ and $x_0 > 0$, and take $y_0 \in R^m$ such that $s_0 = c - A^T y_0 > 0$.*

For $k = 0, 1, 2, \ldots$ until convergence, do (i)–(iii).

(i) *Solve for Δx and Δy from the system*

$$\begin{aligned} (X_k)^{-1} S_k \Delta x - A^T \Delta y &= \theta k (X_k)^{-1} p_k, \\ A \Delta x &= 0, \end{aligned} \tag{3}$$

where $X_k = \mathrm{Diag}(x_k)$ is the positive diagonal matrix whose diagonal elements are the corresponding elements of x_k; $S_k = \mathrm{Diag}(s_k)$;

$$p_k = \frac{e^T z_k}{\rho_k} e - z_k$$

with $z_k = X_k s_k$, $e = (1, 1, \ldots, 1)^T \in R^n$, $\rho_k \geqslant n + \sqrt{n}$; and

$$\theta_k = \frac{\alpha \sqrt{\min(z_k)}}{\| (z_k)^{-1/2} p_k \|}$$

with $\alpha \in (0, 1)$, and $\min(z_k)$ being the smallest element of z_k.

(ii) $\Delta s = -A^T \Delta y$.

(iii) $x_{k-1} = x_k + \Delta x$, $s_{k+1} = s_k + \Delta s$.

It is clear that (3) is of the form of (1) and that solving (3) is the major job in each iteration.

Case 2 (quadratic programming). Similarly, a variety of interior point algorithms for solving the quadratic programming problem

$$\begin{aligned} \min \quad & q(x) = \tfrac{1}{2}x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b, \quad x \geqslant 0 \end{aligned} \tag{4}$$

are described by Y. Ye (1991). Here A, c, x, b are as defined in (2), and Q is an $n \times n$ symmetric positive semidefinite matrix. All those algorithms have the following general form.

ALGORITHM 1.2 (Y. Ye, 1991). *Take* $x_0 \in R^n$ *such that* $Ax_0 = b$ *and* $x_0 > 0$, *and take* $y_0 \in R^m$ *such that* $s_0 = Qx_0 + c - A^T y_0 > 0$.

For $k = 0, 1, 2, \ldots$ *until convergence, do* (i)–(iii).

(i) *Solve for* Δx *and* Δy *from the system*

$$\begin{aligned} \left(Q + D_k^{-1}\overline{D}_k\right)\Delta x - A^T \Delta y &= D_k^{-1}(\lambda_k e - X_k s_k), \\ A\,\Delta x \qquad\qquad &= 0. \end{aligned} \tag{5}$$

Here D_k *and* $\overline{D}_k$ *are positive diagonal matrices,* $\lambda_k \in R$, *and each individual algorithm has a particular way of determining* D_k, $\overline{D}_k$ *and* λ_k.

(ii) $\Delta s = Q\,\Delta x - A^T \Delta y$

(iii) $x_{k+1} = x_k + \Delta x$, $s_{k+1} = s_k + \Delta s$.

Again we mention that (5) is a special case of (1). In this case, the matrix D in (1) is equal to $Q + D_k^{-1}\overline{D}_k$ which is the sum of a positive semidefinite matrix and a positive diagonal matrix.

Case 3 (convex programming). Consider the separable convex nonlinear optimization problem with linear constraints:

$$\begin{aligned} \min f(x) &= c^T x + \sum_{j=1}^{n} w_j x_{(j)} \ln x_{(j)} \\ \text{s.t. } Ax &= b, \quad x \geqslant 0, \end{aligned} \tag{6}$$

where A, c, x, b are as defined in (2), $w = (w_1, w_2, \ldots, w_n)^T \in R^n$ with $w_j \geqslant 0$ for all $j = 1, 2, \ldots, n$, and $x_{(j)}$ is the jth element of x. An interior

point algorithm for solving (6) is proposed by Han et al. (1992), which can be described as the following

ALGORITHM 1.3 (Han et al. 1992). *Take* $x_0 \in R^n$ *such that* $Ax_0 = b$ *and* $x_0 > 0$, *and take* $y_0 \in R^m$ *such that* $s_0 = \nabla f(x_0) - A^T y_0 > 0$.

For $k = 0, 1, 2, \ldots$ *until convergence, do* (i)–(iii).

(i) *Solve for* Δx *and* Δy *from the system*

$$\begin{aligned} \left(\nabla^2 f(x_k) + X_k^{-1} S_k\right) \Delta x - A^T \Delta y &= \theta_k (X_k)^{-1} p_k, \\ A \Delta x \qquad\qquad\quad &= 0, \end{aligned} \tag{7}$$

where θ_k *and* p_k *are the same as defined in* (3).

(ii) *Choose the step size* η.

(iii) $x_{k+1} = x_k + \eta \Delta x$, $y_{k+1} = y_k + \eta \Delta y$, $s_{k+1} = \nabla f(x_{k+1}) - A^T y_{k+1}$.

Since $\nabla^2 f(x_k) = \text{Diag}(X_k^{-1} w)$, (7) is again a special case of (1).

In all the above three algorithms, the major computation in each iteration is to solve an $(n + m) \times (n + m)$ linear system which has the form of (1). Global convergence and polynomial complexity are proved for those algorithms. It turns out that the first n equations do not require exact solutions, while the last m equations have to be solved exactly in order to maintain the feasibility. The same is true for many other interior point methods. For more examples see Lustig et al. (1991), Potra and Shi (1991), Todd and Ye (1990), etc. A similar system also appears in certain potential reduction algorithms for solving linear complementarity problems. For that we refer to Pardalos et al. (1933).

Current implementations solve the system of the form (1) in each iteration by considering the equivalent system

$$AD^{-1}A^T y = -AD^{-1}b, \tag{8}$$

$$x = D^{-1}(A^T y + b). \tag{9}$$

Thus the job is reduced to solving the $m \times m$ system (8). However, (8) has to be solved exactly, since otherwise the error would be entirely passed on to the last m equations of the original system (1). To see this, let y' be an inexact solution of (8), and let

$$AD^{-1}A^T y' = -AD^{-1}b + \eta. \tag{10}$$

From (9) the corresponding solution of x is

$$x' = D^{-1}(A^T y' + b);$$

(10) then implies that $Ax' = \eta$.

Since the exact solution of (8) is needed, iterative methods such as the conjugate gradient method are avoided in solving (8). The typical approach of the current implementation is to use the Cholesky factorization of $AD^{-1}A^T$ to compute the exact solution of (8). Because the matrix D varies from iteration to iteration, one Cholesky factorization is to be done in every iteration. This makes the computation costly and sometimes impractical.

In this paper we propose an iterative method for solving a system of the form (1) in each iteration of those interior point algorithms. It guarantees exact solutions to the last m equations. Instead of computing one Cholesky factorization in each iteration, it computes only one Cholesky factorization to obtain $(AA^T)^{-1}$ at the initial step of the interior point algorithm. We will present the method and its convergence analysis in the next section. In Section 3, some implementational issues are discussed. It is interesting that we may combine this method with another iterative method, such as the conjugate gradient method, to have the advantages of both. In Section 4, some preliminary numerical results are reported.

2. THE ITERATIVE METHOD AND THE CONVERGENCE ANALYSIS

Let us first consider solving (1) with D a positive diagonal matrix. Let $d_1, \ldots, d_n$ be the diagonal elements of D. Let

$$d_{\max} = \max\{d_i;\ i = 1, 2, \ldots, n\},$$

$$d_{\min} = \min\{d_i;\ i = 1, 2, \ldots, n\},$$

and

$$d = \frac{d_{\max} + d_{\min}}{2}.$$

If (x_*, y_*) is the true solution of (1), then we shall have

$$y_* = (AA^T)^{-1} A(Dx_* - b) \tag{11}$$

as well as

$$y_* = (AA^T)^{-1} A[(D - dI)x_* - b], \tag{12}$$

where I stands for the $n \times n$ identity matrix. We shall also have

$$dx_* = A^T y_* + b + (dI - D)x_*. \tag{13}$$

(12) and (13) together imply that

$$dx_* = \left[I - A^T(AA^T)^{-1}A\right][b + (dI - D)x_*]. \tag{14}$$

Our iterative method is based on (11) and (14), and is given as the following algorithm.

ALGORITHM 2.1. *Given* $x^{(1)} \in R^n$ *and* $y^{(1)} \in R^m$ *for* $j = 1, 2, \ldots$ *until convergence do*:

$$y^{(j+1)} = (AA^T)^{-1} A(Dx^{(j)} - b), \tag{15}$$

$$x^{(j+1)} = \frac{1}{d}\left[I - A^T(AA^T)^{-1}A\right][b + (dI - D)x^{(j)}]. \tag{16}$$

First we wish to mention that with Algorithm 2.1

$$Ax^{(j)} = 0 \tag{17}$$

for all $j \geqslant 2$. Thus the solution obtained from Algorithm 2.1 always gives an exact solution to the last m equations of (1). Equation (17) also implies that for all $j \geqslant 2$, (15)–(16) is equivalent to

$$y^{(j+1)} = (AA^T)^{-1} A[(D - dI)x^{(j)} - b], \tag{18}$$

$$x^{(j+1)} = \frac{1}{d}[A^T y^{(j+1)} + b + (dI - D)x^{(j)}]. \tag{19}$$

We now show that $(x^{(j)}, y^{(j)})$ converges to the true solution (x_*, y_*) with any initial point $(x^{(1)}, y^{(1)})$. The matrix norm and the vector norm used in what follows are defined in, for example, Atkinson (1989).

LEMMA 2.1. *If* $A \in R^{m\times n}$, $m < n$, *and* Rank $A = m$, *then* $A^T(AA^T)^{-1}A$ *is symmetric positive semidefinite and*

$$\|I - A^T(AA^T)^{-1}A\|_2 \leqslant 1. \tag{20}$$

Proof. Consider the QR factorization at A^T:

$$A^T = QR, \tag{21}$$

where $Q \in R^{n\times n}$ is orthogonal and $R \in R^{n\times m}$ is upper triangular with Rank $R = m$. Thus R can be expressed as

$$R = \begin{bmatrix} \bar{R} \\ 0 \end{bmatrix} \tag{22}$$

with $\bar{R} \in R^{m\times m}$ upper triangular and invertable. Since $QQ^T = Q^TQ = I$, we have that

$$\begin{aligned} AA^T &= R^TQ^TQR \\ &= R^TR \\ &= [\bar{R}^T \quad 0^T]\begin{bmatrix} \bar{R} \\ 0 \end{bmatrix} \\ &= \bar{R}^T\bar{R}. \end{aligned}$$

Therefore

$$\begin{aligned} A^T(AA^T)^{-1}A &= Q\begin{bmatrix} \bar{R} \\ 0 \end{bmatrix}(\bar{R}^T\bar{R})^{-1}[\bar{R}^T \quad 0^T]Q^T \\ &= Q\begin{bmatrix} \bar{R} \\ 0 \end{bmatrix}\bar{R}^{-1}(\bar{R}^T)^{-1}[\bar{R}^T \quad 0^T]Q^T \\ &= Q\begin{bmatrix} I_{m\times m} \\ 0 \end{bmatrix}[I_{m\times m} \quad 0^T]Q^T \\ &= Q\begin{bmatrix} I_{m\times m} & 0 \\ 0 & 0 \end{bmatrix}Q^T, \end{aligned}$$

which shows that $A^T(AA^T)^{-1}A$ is symmetric positive semidefinite. Furthermore,

$$\begin{aligned}\|I - A^T(AA^T)^{-1}A\|_2 &= \left\| I - Q\begin{bmatrix} I_{m\times m} & 0 \\ 0 & 0 \end{bmatrix} Q^T \right\|_2 \\ &= \left\| Q\begin{bmatrix} 0 & 0 \\ 0 & I_{(n-m)\times(n-m)} \end{bmatrix} Q^T \right\|_2 \\ &\leqslant \|Q\|_2 \left\| \begin{bmatrix} 0 & 0 \\ 0 & I_{(n-m)\times(n-m)} \end{bmatrix} \right\|_2 \|Q^T\|_2 \\ &= \left\| \begin{bmatrix} 0 & 0 \\ 0 & I_{(n-m)\times(n-m)} \end{bmatrix} \right\|_2 \\ &= 1. \end{aligned}$$

■

THEOREM 2.2. *Assume that (x_*, y_*) is the true solution of the system (1) and that the matrix D in (1) is positive diagonal. Let $(x^{(j)}, y^{(j)})$ be produced by Algorithm 2.1 with $(x^{(1)}, y^{(1)})$ an arbitrary initial point. Then for all $j \geqslant 1$,*

$$\|x^{(j+1)} - x_*\|_2 \leqslant \alpha \|x^{(j)} - x_*\|_2, \tag{23}$$

$$\|y^{(j+1)} - y_*\|_2 \leqslant \|(AA^T)^{-1}AD\|_2 \|x^{(j)} - x_*\|_2, \tag{24}$$

where

$$\alpha = \frac{d_{\max} - d_{\min}}{d_{\max} + d_{\min}} \in [0, 1).$$

(23) *and* (24) *thus imply that*

$$(x^{(j)}, y^{(j)}) \to (x_*, y_*).$$

Furthermore, let $\xi^{(j)} = Dx^{(j)} - A^T y^{(j)} - b$. Then for all $j \geqslant 2$

$$\xi^{(j+1)} = (D - dI)(x^{(j+1)} - x^{(j)}), \tag{25}$$

and for all $j \geqslant 3$

$$\|\xi^{(j+1)}\| \leqslant \alpha \|\xi^{(j)}\|. \tag{26}$$

Proof. (14), (16), and (20) lead to

$$\begin{aligned}\|x^{(j+1)} - x_*\|_2 &= \left\| \frac{1}{d}\left(I - A^T(AA^T)^{-1}A\right)(dI - D)(x^{(j)} - x_*)\right\|_2 \\ &\leqslant \frac{1}{d}\|(dI - D)(x^{(j)} - x_*)\|_2 \\ &\leqslant \frac{\max_{1\leqslant i\leqslant n}|d - d_i|}{d}\|(x^{(j)} - x_*)\|_2.\end{aligned}$$

Since $\max_{1\leqslant i\leqslant n}|d - d_i| = (d_{\max} - d_{\min})/2$, (23) is proved. (24) comes immediately from (11) and (15). When $j \geqslant 2$, from (19) we get that

$$(dI - D)x^{(j+1)} = -Dx^{(j+1)} + A^T y^{(j+1)} + b + (dI - D)x^{(j)},$$

which implies (25). Finally, when $j \geqslant 3$,

$$\begin{aligned}\|\xi^{(j+1)}\| &= \|(D - dI)(x^{(j+1)} - x^{(j)})\| \\ &= \left\| \frac{1}{d}(D - dI)\left[I - A^T(AA^T)^{-1}A\right](dI - D)(x^{(j)} - x^{(j-1)})\right\| \\ &= \left\| \frac{1}{d}(D - dI)\left[I - A^T(AA^T)^{-1}A\right](-\xi^{(j)})\right\| \\ &= \frac{1}{d}\left\|(D - dI)\left[I - A^T(AA^T)^{-1}A\right]\xi^{(j)}\right\| \\ &\leqslant \alpha\left\|\left[I - A^T(AA^T)^{-1}A\right]\xi^{(j)}\right\| \\ &\leqslant \alpha\|\xi^{(j)}\|.\end{aligned}$$

■

We now consider the situation when $D = Q + DI$, where Q is symmetric positive semi-definite and DI is positive diagonal. Let $d_1, \ldots, d_n$ be the

diagonal elements of DI; let

$$d_{\max} = \max\{d_i;\ i = 1, 2, \ldots, n\},$$

$$d_{\min} = \min\{d_i;\ i = 1, 2, \ldots, n\},$$

and

$$d = \|Q\|_2 + d_{\max}. \tag{27}$$

Algorithm 2.1 will then be applied with d as defined in (27). All the above convergence results remain true except that in (23) the number α is now changed into

$$\alpha = \frac{d - d_{\min}}{d}. \tag{28}$$

In order to see this, we notice that in this case

$$\|x^{(j+1)} - x_*\|_2 \leqslant \frac{1}{d}\|(dI - D)(x^{(j)} - x_*)\|_2$$

and that

$$\begin{aligned}
&\|(dI - D)(x^{(j)} - x_*)\|_2 \\
&\quad = \|(\|Q\|_2 I - Q)(x^{(j)} - x_*) + (d_{\max} I - DI)(x^{(j)} - x_*)\|_2 \\
&\quad \leqslant \|Q\|_2 \cdot \left\| I - \frac{Q}{\|Q\|_2} \right\|_2 \cdot \|x^{(j)} - x_*\|_2 + (d_{\max} - d_{\min})\|x^{(j)} - x_*\|_2 \\
&\quad \leqslant (\|Q\|_2 + d_{\max} - d_{\min})\|x^{(j)} - x_*\|_2 \\
&\quad = (d - d_{\min})\|x^{(j)} - x_*\|_2.
\end{aligned}$$

Similarly, (26) now still holds with α given in (28). In the next section, the implementation of Algorithm 2.1 with interior point algorithms will be discussed and several conclusions will be addressed.

3. IMPLEMENTATION AND CONCLUSION

For convenience, let us consider implementing the interior point algorithm of the following general form:

ALGORITHM 3.1.

Initial step. Take $x_0 \in R^n$ such that $Ax_0 = b$ and $x_0 > 0$, and take $y_0 \in R^m$ such that $s_0 = \nabla f(x_0) - A^T y_0 > 0$.

Iteration. For $k = 0, 1, 2, \ldots$ until convergence, do:

(3.1.1) Solve for Δx and Δy from the system

$$\begin{aligned} D_k \Delta x - A^T \Delta y &= b_k, \\ A \Delta x \qquad\quad &= 0. \end{aligned} \tag{29}$$

Here $D_k \in R^{n \times n}$ is either a positive diagonal matrix or the sum of a positive diagonal matrix and a symmetric positive semidefinite matrix. D_k and b_k vary from iteration to iteration.

(3.1.2) $x_{k+1} = x_k + \eta \Delta x$, $y_{k+1} = y_k + \eta \Delta y$, $s_{k+1} = \nabla f(x_{k+1}) - A^T y_{k+1}$, where the step size η is often taken as 1, as in Algorithms 1.1 and 1.2.

To implement Algorithm 3.1 using Algorithm 2.1 to solve (29) in each iteration, we add the following to the initial step of Algorithm 3.1:

Compute $(AA^T)^{-1}$ using, for example, the Cholesky factorization.

Instead of computing a Cholesky factorization at each iteration, the above implementation requires only one at the initial step. After $(AA^T)^{-1}$ is computed, Algorithm 2.1 will be used in each iteration to solve (29). The choice of the termination criterion of Algorithm 2.1 depends on the particular interior point algorithms being implemented. Attention should also be paid to the following issues.

It is clear that the number α in (23) [or in (28)] determines the convergence speed of Algorithm 2.1. Let us use α_k to denote the value of α at the kth iteration. If α_k is a small number, then Algorithm 2.1 will converge quickly in solving (29) at this iteration. However, if α_k is very close to 1, the convergence of Algorithm 2.1 at this iteration may be really slow. It is of interest that in this case we may combine Algorithm 2.1 with another method having faster convergence, such as the conjugate gradient method, to

combine the advantages of both. We first consider the equivalent system of (29):

$$AD_k^{-1}A^T \Delta y = -AD_k^{-1}b_k, \tag{30}$$

$$\Delta x = D_k^{-1}(A^T \Delta y + b_k). \tag{31}$$

This is the same as what most current implementations do. But instead of solving (30) exactly, we use an iterative method to get an approximate solution $\Delta y'$. Then we take

$$\Delta x' = D_k^{-1}(A^T \Delta y' + b_k). \tag{32}$$

After that we use $(\Delta x', \Delta y')$ as the initial point and take just one step of Algorithm 2.1 to get a point $(\Delta \tilde{x}, \Delta \tilde{y})$ which will be accepted as the approximate solution of (29) in that iteration. We notice that with $(\Delta x', \Delta y')$, (31) is solved exactly, while in (30) there remains a residual which will cause the same residual in the last m equations of (29) as we mentioned in the first section. With $(\Delta \tilde{x}, \Delta \tilde{y})$ the last m equations of (29) will be solved exactly as we desire, while a residual will remain in the first n equations:

$$D_k \Delta x - A^T \Delta y = b_k. \tag{33}$$

The following theorem shows the connection between those two residuals.

THEOREM 3.1. *Let* $(\Delta x', \Delta y')$ *and* $(\Delta \tilde{x}, \Delta \tilde{y})$ *be as described above; let*

$$AD_k^{-1}A^T \Delta y' = -AD_k^{-1}b_k + \eta \tag{34}$$

and

$$D_k \Delta \tilde{x} - A^T \Delta \tilde{y} = b_k + \xi. \tag{35}$$

Then

(i) $A \Delta x' = \eta$;
(ii) $\Delta \tilde{y} = \Delta y'$;
(iii) $\Delta \tilde{x} = [I - A^T(AA^T)^{-1}A] \Delta x'$;
(iv) $\xi = -D_k A^T (AA^T)^{-1} \eta$.

Before giving the proof of Theorem 3.1, *let us mention that* (*iv*) *can be used to determine whether the residual* η *is to be accepted so that* ξ *will meet*

the termination criterion of Algorithm 2.1. *In other words, (iv) helps in setting up a termination criterion when we solve* (30) *iteratively. Thus our conclusion is: if* α_k *is small, then Algorithm* 2.1 *is used to solve* (29); *otherwise a fast iterative method is used to solve* (30) *and then followed by one step of Algorithm* 2.1. *This way, in both cases we may expect fast convergence and guarantee exact solutions to the last m equations of* (29), *i.e.,* $A\,\Delta x = 0$. *Finally we have the following*

Proof of Theorem 3.1. (i) comes from (32) and (34). (ii) can be seen from (15) and (32). From (16) we have

$$\begin{aligned}
\Delta\tilde{x} &= \frac{1}{d}\left[I - A^T(AA^T)^{-1}A\right]\left[b_k + (dI - D_k)\,\Delta x'\right] \\
&= \frac{1}{d}\left[b_k + (dI - D_k)\,\Delta x' - A^T(AA^T)^{-1}A\,dI\,\Delta x' + A^T\Delta\tilde{y}\right] \\
&= \frac{1}{d}\left[b_k + A^T\Delta y' - D_k\,\Delta x' + dI\,\Delta x' - A^T(AA^T)^{-1}A\,dI\,\Delta x'\right] \\
&= \frac{1}{d}\left[dI\,\Delta x' - A^T(AA^T)^{-1}A\,dI\,\Delta x'\right] \\
&= \left[I - A^T(AA^T)^{-1}A\right]\Delta x'.
\end{aligned}$$

(iii) is thus proved. To obtain (iv), we see that

$$\begin{aligned}
\xi &= D_k\,\Delta\tilde{x} - A^T\Delta\tilde{y} - b_k \\
&= D_k\,\Delta\tilde{x} - D_k\,\Delta x' + D_k\,\Delta x' - A^T\Delta y' - b_k \\
&= D_k(\Delta\tilde{x} - \Delta x') \\
&= -D_k A^T(AA^T)^{-1}\eta.
\end{aligned}$$

■

4. PRELIMINARY NUMERICAL RESULTS

We did some preliminary numerical experiments on an AT & T 3B2-1000 computer using Fortran 77. Four problems taken from Ye (1987) were tested.

TABLE 1
CPU TIME (SEC)

	Prob. 1		Prob. 2		Prob. 3		Prob. 4	
n	A2.1[a]	CFA[b]	A2.1	CFA	A2.1	CFA	A2.1	CFA
40	1.00	1.30	1.23	1.35	1.05	1.26	0.70	0.89
70	1.66	2.46	1.82	2.31	1.65	2.03	0.96	1.31
100	2.03	3.82	2.21	3.67	2.13	2.92	1.10	1.72

[a] Algorithm 2.1.
[b] Cholesky factorization approach.

They are of the form

$$\min \quad f(x)$$

$$\text{s.t.} \quad Ax = b, \quad x \geqslant 0,$$

where $A \in R^{m \times n}$ with $m = 4$ and $n = 3k + 1$ for some integer k. We tested them for $n = 40, 70$, and 100. The objective functions $f(x)$ of the four problems are $\Sigma_{j=1}^{n} - \sqrt{x(j)}$, $\Sigma_{j=1}^{n} - \ln x_{(j)}$, $\Sigma_{j=1}^{n} x_{(j)} \ln x_{(j)}$, and $\Sigma_{j=1}^{n} x_{(j)}^2$. They represent a number of widely used functions in economics, physics, and engineering. Those problems can be solved using Algorithm 1.3 (see Potra and Shi, 1991), and the third problem is indeed of the form (6) with $c = 0$ and $w = e$. Algorithm 1.3 was thus used, and all the parameters as well as the step size were chosen the same as those in Han et al. (1992). The algorithm was terminated when the duality gap $x_k^T s_k$ was less than 10^{-11}. We compared Algorithm 2.1 with the typical Cholesky factorization approach for solving (29). The initial points for Algorithm 2.1 were set as $\Delta x = 0$ and $\Delta y = 0$. Table 1 shows that the CPU time used by Algorithm 1.3 with Algorithm 2.1 is, in all the cases, less than that used with the Cholesky factorization approach.

The author would like to thank the referee for valuable comments and suggestions.

REFERENCES

Atkinson, Kendall E. 1989. *An Introduction to Numerical Analysis*, Wiley.

Han, Chi-Geun, Pardalos, Panos M., and Ye, Yinyu. 1992. Implementation of interior point algorithms for some entropy optimization problems, *Optim. Methods and Software* 1:71–80.

Kortanek, K. O., Potra, F. A., and Ye, Y. 1991. On some efficient interior point methods for nonlinear convex programming, *Linear Algebra Appl.* 152:169–189.

Lustig, I. J., Marsten, R. E., and Shanno, D. F. 1991. The interaction of algorithms and architectures for interior point methods, Preprint.

Pardalos, P. M., Ye, Y., Han, C.-G., and Kalinski, J. 1993. Solution of p_0-matrix linear complementary problems using a potential reduction algorithm, *SIAM J. Matrix Anal. Appl.* 14:1048–1060.

Potra, F. A. and Shi, Y. 1991. On an Efficient Interior Point Method for Solving Entropy Optimization Problems, Report on Computation Mathematics 23, Univ. of Iowa.

Tapia, R., Zhang, Y., Saltzman, M., and Weiser, A. 1991. The predictor-corrector interior-point method as a composite Newton method, Preprint.

Todd, M. J. and Ye, Y. 1990. A centered projective algorithm for linear programming, *Math. Oper. Res.* 15:508–529.

Ye, Y. 1987. Interior Algorithms for Linear, Quadratic, and Linear constrained Convex Programming. Ph.D. Thesis, Dept. of Engineering and Economic Systems, Stanford Univ.

Ye, Y. 1991. Interior-point algorithms for quadratic programming, in *Recent Developments in Mathematical Programming* (S. Kumar, Ed.), Gordon & Breach, Philadelphia.

Ye, Y. 1992. On the finite convergence of interior-point algorithms for linear programming, *Math. Programming* 57:325–335.

Ye, Y., Güler, O., Tapia, R. A., and Zhang, Y. 1991. A quadratically convergent $o(\sqrt{n}\,L)$-iteration algorithm for linear programming, *Math. Programming*, to appear.

Ye, Y. and Todd, Michael. 1990. Containing and shrinking ellipsoids in the path-following algorithm, *Math. Programming* 47:1–9.

Received 30 September 1993; final manuscript accepted 27 December 1993